D4A Magazine
مجلة منتدى دلفي للعرب
Issue 03 - August 2010
العدد رقم : 03 لشهر أغسطس 2010



فهـــرس العدد



برامج الحماية و الــ False positive

أصبحت مصادفة رسائل التحذير الخاطئة التي تظهرها برامج الحماية هاجس يزرع الشك في مصداقية البرامج، و تخلط المفاهيم على المستعمل العادي و كيف و إن كان من يظهر الرسالة هو برنامج حماية مصدر ثقة نصب على الجهاز لكي يحمي المستعمل من البرامج الضارة.

السؤال هو كيف استطيع أن افرق بين برنامج ضار حقيقي و بين برنامج نظيف تم تصنيفه ؟

الإجابة معقدة، حيث أن الحل يكمن في الفحص البشري للبرنامج المشكوك فيه من طرف مخابر برامج الحماية لكي يتم حذفه من قائمة البرامج الضارة في حالة تصنيف خاطئ.

لماذا يصبح برنامجي مصنف ؟:

1. التشابه في السلوك مع برنامج ضار.
2. استعمال برامج ضغط/حماية الملفات التنفيذية.
3. تشفير أجزاء من الأوامر لحمايتها مثل خوارزميات التسجيل.
4. مواقع الفحص على الانترنت ساهمت أيضا في انتشار التصنيفات الخاطئة، عند فحص برنامج على موقع virustotal مثلا فإن هذا الأخير بناءا على اتفاقية أبرمت مع شركات برامج الحماية يقوم بإرسال نسخة من البرنامج المفحوص إليها.

حالة False positive:

فحص شركة 01	فحص شركة 02	فحص شركة 03	فحص شركة 04	فحص شركة 05	فحص شركة 06
تتيجة سلبية	تتيجة سلبية	تتيجة سلبية	تتيجة ايجابية	تتيجة سلبية	تتيجة سلبية
يتم إرسال نسخة من البرنامج المفحوص مع تقرير الفحص	يتم إرسال نسخة ... مع تقرير الفحص	يتم إرسال نسخة ... مع تقرير الفحص		يتم إرسال نسخة ... مع تقرير الفحص	يتم إرسال نسخة ... مع تقرير الفحص
تقوم الشركة بإضافة البرنامج إلى قاعدة البيانات	تقوم الشركة بإضافة...	تقوم الشركة بإضافة...		تقوم الشركة بإضافة...	تقوم الشركة بإضافة...
False positive	False positive	False positive	False positive	False positive	False positive

حالة كشف صحيح

فحص شركة 01	فحص شركة 02	فحص شركة 03	فحص شركة 04	فحص شركة 05	فحص شركة 06
نتيجة سلبية	نتيجة سلبية	نتيجة سلبية	نتيجة ايجابية	نتيجة سلبية	نتيجة سلبية
يتم إرسال نسخة من البرنامج المفحوص مع تقرير الفحص	يتم إرسال نسخة ... مع تقرير الفحص	يتم إرسال نسخة ... مع تقرير الفحص		يتم إرسال نسخة ... مع تقرير الفحص	يتم إرسال نسخة ... مع تقرير الفحص
تقوم الشركة بإضافة البرنامج إلى قاعدة البيانات	تقوم الشركة بإضافة...	تقوم الشركة بإضافة...		تقوم الشركة بإضافة...	تقوم الشركة بإضافة...
كشف حقيقي	كشف حقيقي	كشف حقيقي	كشف حقيقي	كشف حقيقي	كشف حقيقي

حالة نتيجة سلبية

فحص شركة 01	فحص شركة 02	فحص شركة 03	فحص شركة 04	فحص شركة 05	فحص شركة 06
نتيجة سلبية	نتيجة سلبية	نتيجة سلبية	نتيجة سلبية	نتيجة سلبية	نتيجة سلبية
لا يتم إرسال التقرير					

و ليس المبرمج الهاوي فقط من يكون ضحية التصنيف الخاطئ بل لاحظنا شركات حماية عريقة مثل Panda Labs **يتم تصنيف احد تحدياتها على أنه برنامج ضار، راجع موضوع تحدي** Panda Challenge 2010 **على موقعهم الرسمي**.

الكاتب: إدارة المنتدى

إعادة الاستخدام في دلفي ـ بقلم خالد شقروني

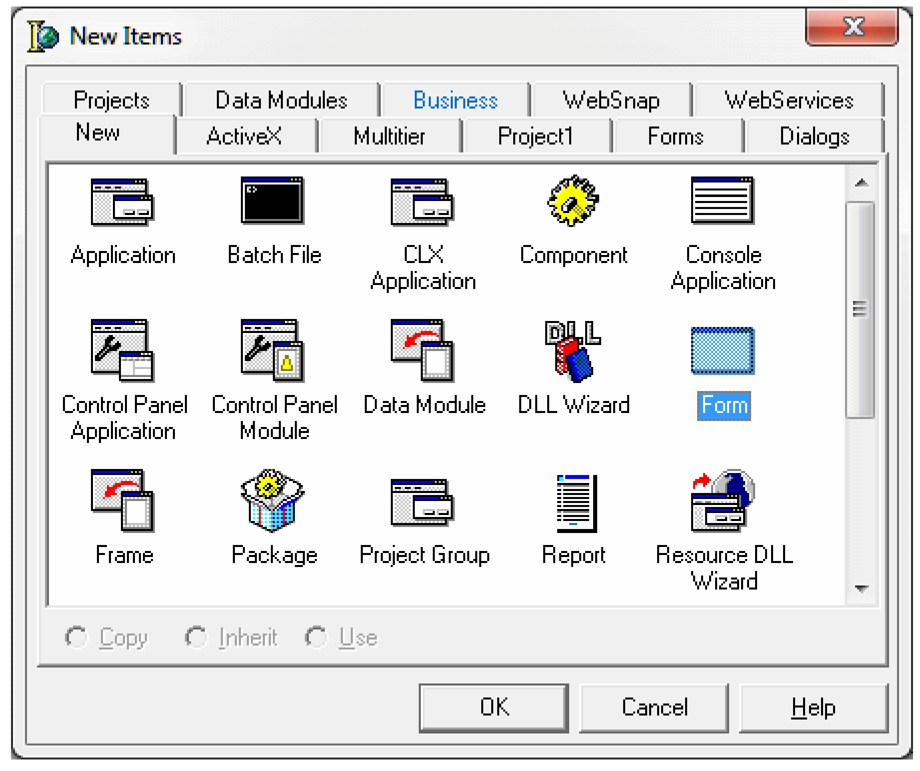
الجزء الثالث – المستودعات

في هذا الجزء نستكمل سلسلة مقالات حول إعادة الاستخدام في دلفي، و الذي بدأناه بالقوالب Templates في الجزء الأول، و الإطارات Frames في الجزء الثاني. و في هذا الجزء الثالث سنتحدث عن مستودع القوالب Repository في دلفي.

المستودع Repository في دلفي هو مكان تقوم فيه دلفي بتخزين قوالب كائنات أو صنفيات Calsses مختلفة (نماذج شاشات، مشاريع، برامج wizard) سابقة الإعداد لاستخدامها لاحقا من قبل المبرمج كأساس لعناصر جديدة في برامجه. في الواقع عندما نطلب من دلفي إنشاء مشروع جديد، أو إنشاء شاشة أو وحدة جديدة فإن دلفي تقوم باستدعاء قالب سابق الإعداد من المستودع ذاته.

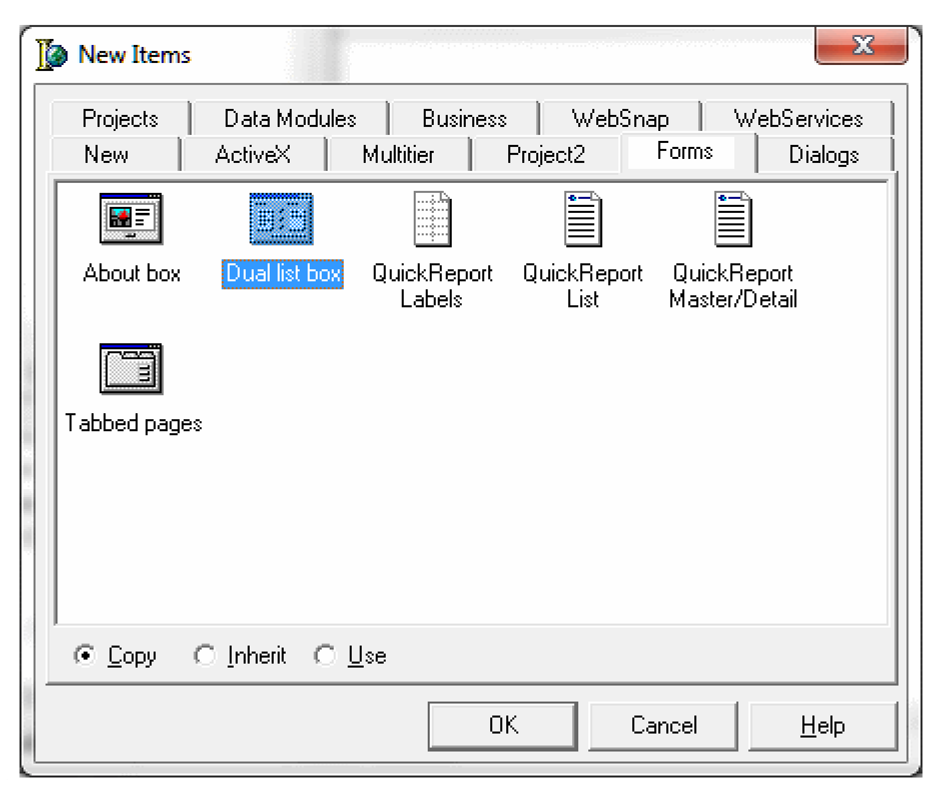
لنأخذ لمحة سريعة ومختصرة لكيفية استخدام المستودع.

يتم إستدعاء قالب من المستودع من خلال لائحة الأوامر: ..File| New| Other (قد تختلف سلسلة الأوامر من نسخة لأخرى) فتظهر شاشة New Item كما في الشكل 01، عارضة في صفحات مبوبة أنواع القوالب التي سينبثق منها العنصر الجديد الذي نود إدراجه في مشروعنا.



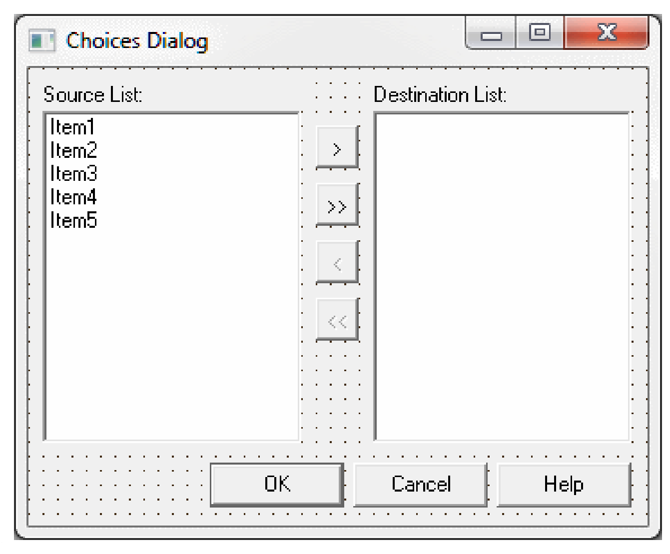
الشكل 01

نختار الصفحة Forms ومنها نختار القالب Dual List Box كما في الشكل 02



الشكل 02

عندها تقوم دلفي بإضافة شاشة جديدة بنفس بنية القالب الذي اخترناه كما في الشكل 03



الشكل 03

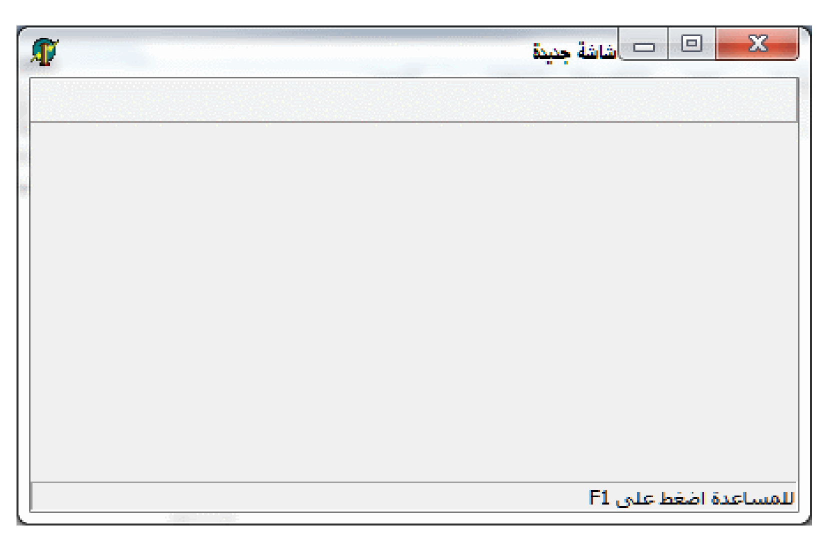
الشاشة الجديدة هي شاشة سابقة الإعداد تحتوي على عناصر الاستخدام من أزرار و قوائم، كما أنها تحتوي على التوليف Code **اللازم لنقل عناصر القوائم من قائمة لأخرى.**

كما لاحظنا، نستطيع أن نستدعي عنصرا سابق الإعداد (مثل الشاشة السابقة) ويكون أساسا قابلا للتعديل والإضافة، وتوظيفه في برامجنا بدلا من استدعاء شاشة فارغة والعمل عليها من جديد.

كيف نقوم بتخزين قالب في المستودع

لنأخذ مثالا عمليا لكيفية تخزين قالب شاشة form **في المستودع، لنفترض أننا نريد للشاشات في برامجنا أن تكون بصيغة خاصة وموحدة، كأن تكون فيها** TPanel **أعلى الشاشة و** TStatusbar **أسفلها، وأن تكون خاصية الاتجاه من اليمين لليسار**bdRightToLeft**، كما أن نوع الخط على مستوى الشاشة يكون نوع** Tahoma **عربي و خاصية** AutoScroll **تكون سالبة. وحتى نوفر على أنفسنا عناء تكرار هذه العمليات كل مرة، نريد حفظها كقالب يتم استخدامه كلما احتجنا لمثل هذه الشاشة.**

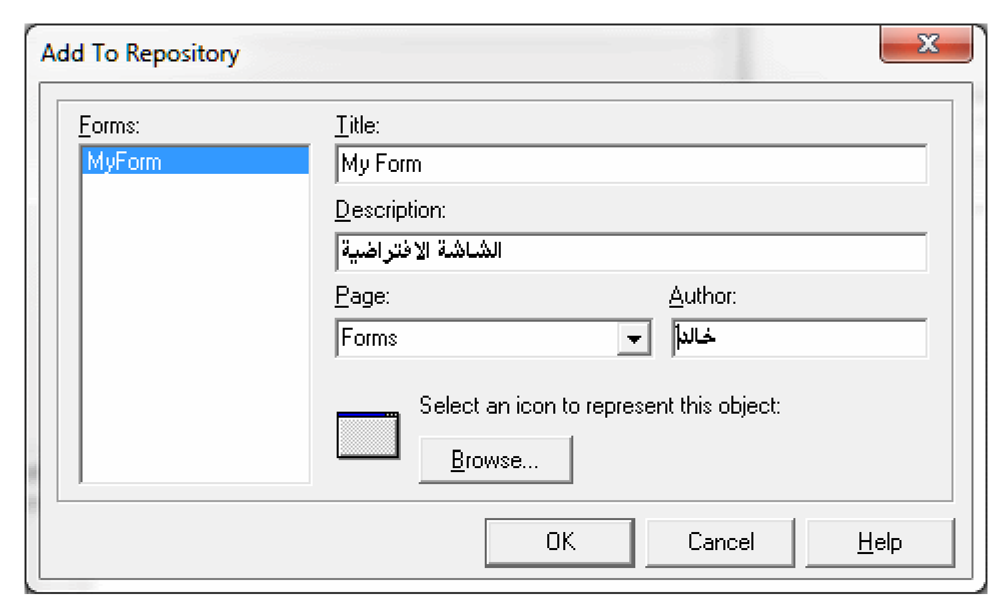
نقوم بإعداد شاشة بالخصائص المطلوبة مثل الشكل 04. نقوم بتسمية نموذج الشاشة بإسم MyForm **ونحفظها بإسم** fMyForm **في مجلد نختاره.**



الشكل 04

الآن نريد من دلفي أن تجعل من نموذج الشاشة هذا قالبا Template **يمكن استدعاؤه لاحقا من المستودع** Repository.

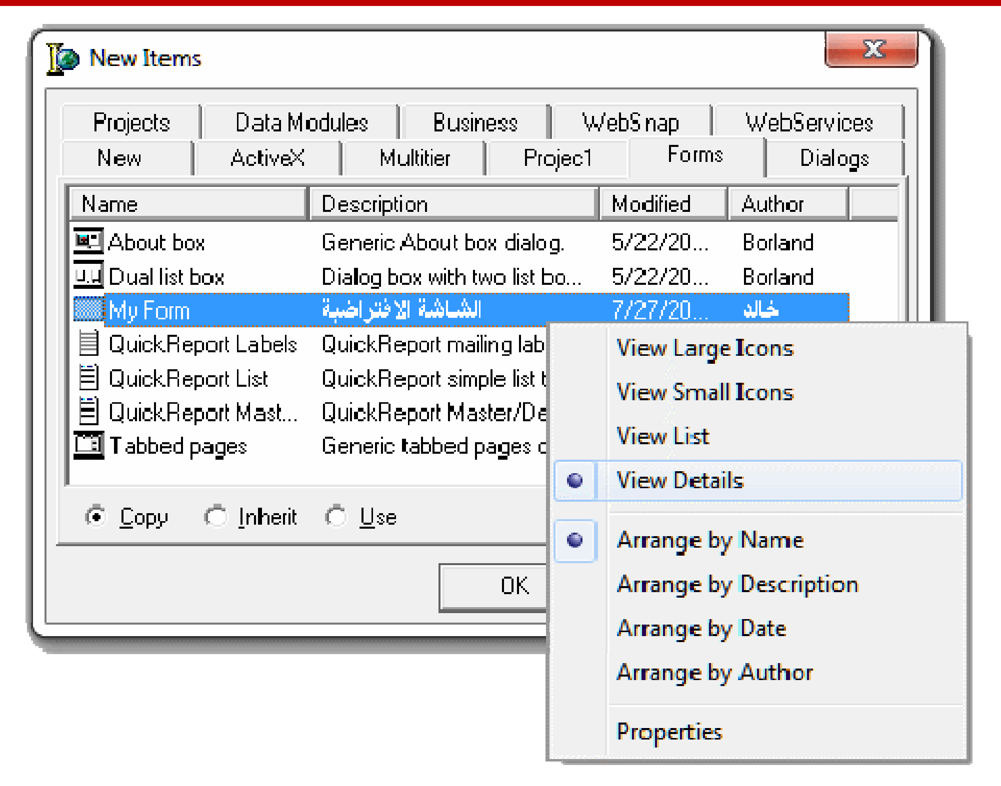
من خلال النقر على الزر الأيمن للفأرة في النموذج، ومن خلال لائحة الأوامر الفرعية نجد الأمر: Add to Reository... ، بإختيار هذا الأمر تظهر لنا شاشة الحوار الخاصة بالإضافة إلى المستودع، نقوم بملء البيانات فيها كما في الشكل 05



الشكل 05

البيانات تتضمن عنوانا نضعه لهذا القالب ووصفا له وإسم من قام بإعداد هذا القالب، ثم إسم الصفحة في المستودع التي سيندرج تحتها، بالنسبة لإسم الصفحة يمكن اختيارها من قائمة الصفحات الموجودة فعلا بالمستودع، أو نضع إسم لصفحة جديدة. أيضا هناك بيانات خاصة بملف الأيقونة icon **التي سترمز لهذا القالب في المستودع، بالامكان إختيار ملف أيقونة (**ico.*) **أو اعتماد الأيقونة الافتراضية الظاهرة.**

الآن لو رجعنا إلى المستودع Repository **(عن طريق الأمر:** File| New| Other...**) وذهبنا إلى صفحة** Forms **سنجد القالب** My Form **الذي أنشأناه مدرجا ضمن القوالب الأخرى، وجاهز للإستخدام.**



الشكل 06

إذا أردنا عرض معلومات أكثر عن هذه القوالب؛ فمن خلال لائحة الأوامر الفرعية (الزر الأيمن) نختار الأمر View Details ليتم عرض تفاصيل كل قالب كما في الشكل 06.

يمكننا من الآن استخدام القالب الجديد الذي كوّنّاه بالضغط على زرّ موافق Ok فتقوم دلفي بإعطائنا نموذج شاشة جديدة عبارة عن نسخة من القالب الأصلي. ويمكننا أن نعيد هذا الأمر أكثر من مرة.

استخدام قوالب المستودع

الآن لنستكشف كيفية استخدام قالب من المستودع بتمعن أكثر.

لنقم بإنشاء مشروع جديد، ثم طلب إضافة عنصر جديد من المستودع (File| New| Other...)، نختار صفحة Forms ونختار قالبنا الذي قمنا بتكوينه سابقا My Form. الآن لو تفحصنا شاشة الإضافة من المستودع سنرى ثلاث خيارت أسفل الشاشة (راجع الشكل 06) وهي Copy نسخ، و Inherit توريث، و Use استخدام. أي أن إضافة نموذج الشاشة My form بإستخدام هذا القالب يمكن أن يتم بخيارات ثلاث:

1ـ النسخ Copy، و يعني أن دلفي ستقوم بإعطائنا نموذجا جديدا يضاف لبرنامجنا هو عبارة عن نسخة طبق الأصل من النموذج القالب. وبالتالي فإن صنفية class (نوع) النموذج الجديد ستكون منبثقة من نفس الصنفية التي انبثق منها القالب.

2ـ توريث Inherit، ويعني أن النموذج الجديد سيكون منبثقا من صنفية class القالب.

3ـ استخدام use، ويعني أن برنامجنا سيستخدم ملف القالب مباشرة.

بتفصيل أكثر:

القالب الأصلي كأي نموذج شاشة آخر هو صنفية class منبثقة من الصنفية TForm وتعريفها كالتالي:

```
type
  TMyForm = class(TForm)
```

(1) لو انشأنا نموذج شاشة جديدة باستخدام القالب مع خيار النسخ Copy فسيكون التعريف (تضعه دلفي آليا) كالتالي:

```
type
  TMyForm1 = class(TForm)
```

أي أنها نسخة طبق أصل القالب، وفيها نفس كل التعريفات و الأكواد التي بالقالب الأصلي. وبما أنها نسخة من القالب، فستكون مستقلة عن القالب و لا تتبعه، و بالتالي أي تغيير في القالب الأصلي سوف لن يؤثر على هذه النسخة، كما أنه يمكننا تعديل وإلغاء ما شئنا من عناصر وأكواد في هذه النسخة.

(2) الخيار الثاني هو التوريث، فإذا أضفنا نموذج شاشة جديد باستخدام القالب مع خيار Inherit فإن النموذج الجديد سيكون مولدا من القالب الأصلي وتعريفه كالتالي:

```
type
  TMyForm1 = class(TMyForm)
```

لذلك سنلاحظ أن النموذج الجديد خاليا من أية تعريفات للعناصر الموجودة، كما أنه خاليا من أية أكواد. لأن عناصر وأكواد نموذج الشاشة هذا معرفة في القالب الأصلي الذي انبثق منه وورث منه كل خصاصه. والذي قامت دلفي بإضافة إشارة لملفه في قسم uses.

```
uses
  Windows, Messages, SysUtils, Variants,
  Dialogs, FMYFORM, ComCtrls, ExtCtrls;

type
  TMyForm1 = class(TMyForm)
  private
```

الشكل 07

وبهذا لا يمكننا إلغاء عنصر موجود فيها (يمكننا تغيير خصائصه)، كما أن أي تغيير قد يطرأ لاحقا على القالب الأصلي سوف ينعكس تأثيره على جميع النماذج المنبثقة منه. وينطبق عليه نفس ما تم شرحه عند الحديث عن الإطارات Frames.

(راجع الجزء الثاني من هذه المقالة والخاص باستخدام الإطارات، مجلة دلفي للعرب العدد 2)

(3) الخيار الثالث هو الاستخدام Use**، ويعني أننا سنستخدم ملف القالب مباشرة. وبالتالي فإن أي تغيير يتم على هذا القالب سوف ينعكس على كل نماذج الشاشات التي استخدمت هذا القالب بالوراثة، سواء في مشروعنا الحالي أو في مشاريع أخرى. لذلك يجب الحذر عند اعتماد هذا الخيار.**

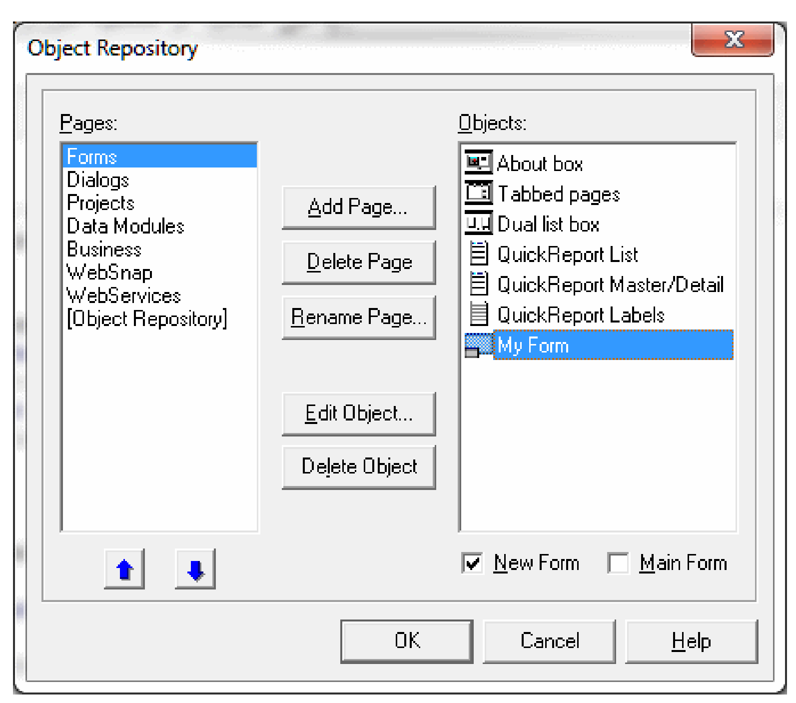
ملخص الخطوات السابقة:

قمنا بإعداد نموذج شاشة form **وفق الخصائص التي نرغبها (اتجاه، خط، عناصر مضافة)، وقمنا بحفظها ، و إضافتها كقالب إلى المستودع عن طريق الأمر الفرعي** Add to Repository. **وعندما نريد إضافة شاشة جديدة في أي مشروع باستخدام هذا القالب؛ نطلب الأمر** File| New| Other **ونختار القالب من المستودع، مع خيار أن يكون النموذج الجديد نسخة مستقلة من القالب أو صنفية موروثة منه.**

الآن سنستكشف أبعادا جديدة في المستودع Repository.

ماذا لو أردنا أن نجعل من دلفي يقوم تلقائيا باستدعاء الشاشة السابقة عند طلب شاشة جديدة، دون أن نضطر إلى الذهاب إلى المستودع واختيارها من هناك؟ بما أننا قد قمنا بإعداد القالب السابق ليكون كأساس لمعظم شاشات مشاريعنا، فإنه من الجميل أن يكون القالب السابق هي الشاشة الافتراضية عند طلب نموذج شاشة جديد.

الأمر سهل. من خلال قائمة الأوامر الرئيسية في دلفي نختار الأمر.. Tools| Repository فتظهر لنا شاشة إدارة قوالب كائنات المستودع Object Repository والتي تتيح لنا إدارة محتويات المستودع، كما في الشكل 08.



الشكل 08

بعد اختيارنا للقالب My Form كما هو ظاهر في الشكل أعلاه، سنجد أسفل شاشة إدارة المستودع خيار New Form أي شاشة جديدة، و هذا يعني أننا إذا فعّلنا هذا الخيار فإن هذا القالب سيكون أساس أية شاشة جديدة في دلفي. يمكننا تفعيل هذا الخيار واعتماده. و سنلاحظ أنه كلما أردنا إضافة form جديدة ستكون بنفس الخصائص التي حددنا بالقالب.

إذا أردنا إختيار قالب آخر ليكون شاشة النموذج الافتراضية، يمكن العودة إلى شاشة إدارة المستودع وإختيار قالب لشاشة أخرى. سنلاحظ أن شاشة إدارة المستودع لا تسمح إلا بإختيار قالب واحد ليكون القالب

الإفتراضي لأية شاشة جديدة، و سيقوم تلقائيا بإزالة الاختيار عن القالب الأول. (لاحظ تغيرشكل أيقونة القالب عند تفعيل أو إزالة تفعيل الخيار).

بجانب خيار New Form في شاشة إدارة المستودع يوجد خيار Main Form أي نموذج الشاشة الرئيسية ـ راجع الشكل 08 – إذا تم تفعيل هذا الخيار، فإن دلفي ستقدم هذه الشاشة كشاشة رئيسية كلما أنشأنا مشروعا جديدا.

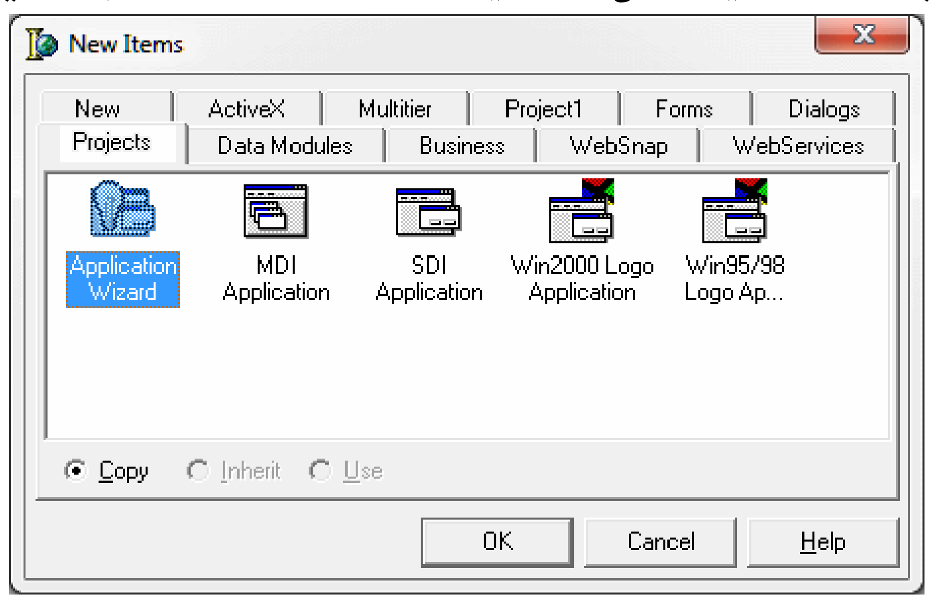
تنبيه: يجب توخّي الحذر عند التعامل مع هذه الخيارات لأنها المسؤولة عن نماذج الشاشات الافتراضية في دلفي.

تحديد قوالب المشاريع:

كما توجد قوالب لكائنات منفردة مثل نماذج الشاشات forms؛ توجد أيضا قوالب لمشروعات مكتملة. عندما نطلب من دلفي إنشاء تطبيق جديد File| New| Application (كما في دلفي 6)، فإن دلفي تقدم لنا مشروعا جديدا مكونا من شاشة form جديدة وحيدة. يمكننا عن طريق المستودع أن نحدد أن المشروع الجديد يكون بقالب مشروع سابق الإعداد.

لو تأملنا شاشة اختيار القوالب من المستودع سنجد ضمن الصفحات صفحة بإسم Project مشاريع، وبها مجموعة من قوالب المشاريع التي يمكن إختيار إحداها لتكون أساسا لتطبيق جديد. أنظر الشكل 09.

يمكننا اختيار إحداها، وستقوم دلفي بنسخ محتويات كافة ملفات هذا المشروع في المجلد الذي نحدده، ليكون أساسا لتطبيق جديد. لاحظ أن خيار النسخ هو الوحيد المتوفر عند استخدام قوالب المشاريع.



الشكل 09

إضافة مشروع كقالب للمستودع:

نحن يمكننا أيضا تحديد مشروع كقالب و تخزينه في المستودع.

يمكن أن نبدأ مشروعا جديدا تكون شاشته الرئيسية نفس الشاشة التي حددناها سابقا، و نضيف إليها إذا أردنا شاشات أخرى مثل شاشة About عن البرنامج ، و رسالة تنبيه Message Box تظهر عند غلق الشاشة الرئيسية للتأكيد على غلق البرنامج. وبعد حفظ المشروع و ملفاته. نطلب من دلفي تخزينه كقالب في المستودع و ذلك عن طريق الأمر Project| Add to Repository فيقوم دلفي بعرض شاشة نضع فيها بيانات هذا القالب. و سيكون جاهزا لإعادة الاستخدام لاحقا.

ملاحظات عامة عن المستودع:

- **في المستوع يمكن رؤية صفحة بنفس عنوان مشروعنا الحالي بها قائمة بجميع نماذج الشاشات المستخمة في المشروع، و هي بالتالي تتيح لنا إضافة شاشة جديدة لمشروعنا تكون منبثقة و مورثة من إحدى هذه الشاشات.**
- **العناصر التي ضمن صفحة New تكون دائما بالنسخ و لاتتوفر فيها خاصية التوريث، كذلك القوالب الخاصة بالمشاريع.**
- **القوالب التي تأتي جاهزة مع دلفي مخزنة في المجلد Objrepos في مسار مكان تثبيت دلفي.**
- **الملف الذي فيه التعريفات الخاصة بالمستودع موجود في مسار تثبيت دلفي في:** Bin\delphi32.dro
- **يمكن لفريق عمل أن يتشارك في مستودع موحد عبر الشبكة، و يتم تحديد المجلد الذي سيكون فيه المستودع و ملف تعريفاته من خلال لائحة الأوامر: Tools|Environment Options ثم صفحة Preferences في قسم Shared repository**

قريبا، العدد المخصص لشهر رمضان

مواضيع خفيفة غير سطحية سهلة الهضم.



مجلة منتدى دلفي للعرب، منكم واليكم

قواعد البيانات – بقلم Kachwahed

لغة الاستعلامات البنيوية – أمثلة وتطبيقات (الجزء الثاني)

Structured Query Language - Examples and Training (Part 2)

تحدثنا في الموضوع السابق عن مميزات لغة SQL، **وذكرنا أنها لغة تصريحية** (Declarative)، **خلافا للغات البرمجة الأخرى النمطية** (Procedural)، **مما يجعلها سهلة الفهم والاستعمال ولا تتطلب أية معارف برمجية مسبقة بلغة أخرى.**

وسنعرض في هذا الموضوع إلى أهم أوامر لغة SQL **مع ذكر بعض أهم الاختلافات بين أشهر المعايير المستخدمة في لغة** SQL.

القواعد النحوية في SQL:

رأينا سابقا بعض القواعد النحوية لجمل SQL **ويمكن تلخيصها فيما يلي:**

- **لا فرق بين الأحرف الكبيرة** (Upper Case) **والأحرف الصغيرة** (Lower Case)
- **الفراغات (المسافات) لا يعتد بها، ولا فرق بين مسافتين (أو أكثر) ومسافة واحدة، ونفس الأمر فيما يخص رمز السطر** (#13) [1]
- **تنتهي كل جملة في** SQL **برمز القاطعة** (;)، **ووجود القاطعة في وسط الجملة يعني نهايتها.**

تعليمات معالجة البيانات (DML) Data Manipulation Language:

- **الصيغة العامة الشهيرة للاستعلام عن بيانات جدول:**

```
SELECT ColumsName FROM TableName;
```

SELECT: **كلمة محجوزة في** SQL **وتعني اختر، انتق، حدد...**

ColumsName: **اسم الحقل (أو الحقول) التي تود تحديدها.**

FROM: **كلمة محجوزة في** SQL **وتستخدم لتحديد الجدول (أو الجداول) التي تحتوي على الحقل (الحقول) المحددة.**

TableName: **اسم الجدول (أو الجداول أو تركيبة من الجداول) التي تحوي الحقول.**

- **لتحديد عدة حقول يجب الفصل بينها بفاصلة، مثال:**

```
SELECT FirstName, LastName FROM Employee;
```

[1] باستثناء مسافات الفراغ في أسماء الأغراض أو المسارات طبعا.

- لتحديد كافة الحقول نستخدم الرمز * في محل أسماء الحقول، مثال:

```
SELECT * FROM Employee;
```

- لعرض أحد الحقول باسم مستعار نستخدم كلمة AS متبوعة بالاسم المستعار، مثال:

```
SELECT FirstName AS EmployeeName, LastName FROM Employee;
```

وهنا اختلاف صغير بين Paradox 7 (المعيار SQL-92) حيث يمكن حذف عن AS، مثال:

```
SELECT FirstName EmployeeName, LastName FROM Employee;
```

وهو ما لا يمكن في Access (المعيار SQL-89) ويعد خطأ نحويا. بينما يمكن هنا استخدام اسم مستعار يحتوي فراغات ونحوها مع إدراجه بين شالتين أو مزدوجتين أو معقوفتين، مثال:

```
SELECT FirstName AS 'Employee Name', LastName FROM Employee;
```

أو

```
SELECT FirstName AS "Employee Name", LastName FROM Employee;
```

أو

```
SELECT FirstName AS [Employee Name], LastName FROM Employee;
```

وهذا ما لم يمكن ممكنا في Paradox 7 (المعيار SQL-92).
في SQL Server (المعيار SQL-99) العبارات السابقة جميعها صحيحة J .

- يمكن اتخاذ اسم مستعار للجدول، وبالتالي يمكن كتابة الجملة السابقة بكل هذه الصيغ:

```
SELECT Employee.FirstName, Employee.LastName FROM Employee;
SELECT Emp.FirstName, Emp.LastName FROM Employee AS Emp;
SELECT Emp.FirstName, Emp.LastName FROM Employee Emp;
```

من الضروري تحديد اسم الجدول إذا وجدت عدة جداول تحمل أسماء حقول متشابهة.

- لاستخراج البيانات دون تكرار نضيف كلمة DISTINCT، مثال:

```
SELECT DISTINCT FirstName FROM Employee;
```

- لترتيب البيانات تصاعديا نستخدم الأمر ORDER BY:

```
SELECT FirstName, LastName FROM Employee ORDER BY FirstName;
```

هذا لترتيب البيانات تصاعديا ASC. ولترتيب السجلات تنازليا نضيف DESC، هكذا:

```
SELECT FirstName, LastName FROM Employee ORDER BY FirstName DESC;
```

يمكن عرض البيانات في مستويات من الترتيب، مثال:

```
SELECT EmpID, FirstName FROM Employee ORDER BY EmpID, FirstName;
```

حيث يجب إدراج حقل الترتيب (EmpID في المثال) ضمن قائمة الحقول المحددة.

- يمكن إدراج شرط منطقي (مساواة، أكبر من، يختلف عن...) في الاستعلام، مثال:

```
SELECT EmpID, FirstName FROM Employee WHERE EmpID = 10;
```

ويمكن هنا استخدام أي من العلامات = أو < أو > أو <= أو >= أو <> .

لتحقيق مجموعة من شرط (أو شروط) أو/و تحقيق انتفاء أحدها (أو بعضها)، يمكن استخدام المعاملات المنطقية AND و OR و NOT، مثال:

```
SELECT EmpID, FirstName, Age, Married FROM Employee WHERE EmpID < 100 AND Age <
60 AND NOT Married;
```

الحقل Married يحمل قيمة منطقية (True أو False)، الجملة السابقة صحيحة لدى البعض (Access) وخاطئة لدى آخرين، حيث يجب تحديد قيمة الحقل: Married = False.

يستحسن استخدام الأقواس عند وجود عدة شروط لرفع اللبس عن المترجم، فمثلا هذه:

```
SELECT EmpID, Age, FirstName FROM Employee WHERE (EmpID < 100) AND NOT (Married
= True OR Age < 60);
```

تختلف تماما عن هذه:

```
SELECT EmpID, Age, FirstName FROM Employee WHERE (EmpID < 100) AND NOT (Married
= True) OR (Age < 60);
```

وهذه الأخيرة هي التي ينفذها مترجم SQL عند عدم استخدام الأقواس.

- كما يمكن استخدام المعامل LIKE مثل الآتي:

```
SELECT * FROM Employee WHERE FirstName LIKE 'Omar';
```

وطبعا قيم الحقول النصية يجب أن تكتب بين شالتين '' أو مزدوجتين "" .

يمكن استخدام المعامل LIKE مع الرمز % للاستعلام عن كلمة مشابهة، مثال:

```
FROM Employee WHERE FirstName LIKE 'A%';*SELECT
```

يعني هذا المثال عرض جميع الأسماء التي تبدأ بالحرف A. مثال آخر:

```
FROM Employee WHERE FirstName LIKE '%OU%';*SELECT
```

هذا المثال يبحث عن الأسماء التي تحتوي على حرفي OU في وسط الكلمة. للبحث في آخر الكلمة نجعل الرمز % في أول كلمة البحث: '%A'.

يعني الرمز ٪ عددا من الحروف، لتحديد عدد من الحروف نستخدم الرمز _، مثال:

```
FROM Employee WHERE FirstName LIKE '_OU%';*SELECT
```

تستعلم هذه الجملة عن الأسماء التي تبدأ بحرف ثم OU ثم مجموعة من الأحرف.

يمكن البحث عن مجال من الأحرف، مثالا للبحث عن الأسماء التي تبدأ بحرف من أحرف المجال [A-L] نكتب:

```
FROM Employee WHERE FirstName LIKE '[A-L]%';*SELECT
```

هذه الأخيرة غير موجودة في Paradox 7 (المعيار SQL-92).

كما يمكننا البحث عن أحد الأسماء التي لا تبدأ بحرف (أو مجال من الأحرف) معين:

```
FROM Employee WHERE FirstName LIKE '[^B]%';*SELECT
```

هذه الأخيرة غير ممكنة في Paradox 7 و Access وممكنة في SQL Server (المعيار SQL-99).

- يمكن أيضا استخدام المعامل BETWEEN (مع إضافة AND) لتحديد مجال الشرط كالآتي:

```
SELECT EmpID, FirstName FROM Employee WHERE Age BETWEEN 30 AND 60;
```

وليس من الضروري عند استخدام LIKE أو BETWEEN إدراج حقل الشرط في ضمن الحقول المحددة، لاحظ في الجملة السابقة أن الحقل Age غير محددة ضمن SELECT.

- يمكن استخدام المعامل IN مع تحديد مجموعة من القيم، مثال:

```
SELECT EmpID, FirstName FROM Employee WHERE Age IN (25, 26);
```

- تكتب التواريخ بين علامتين # في Access أو شالتين " في Paradox 7 وSQL Server، ويمكن استخدام مزدوجتين للهلّه في Paradox 7 أيضا، أمثلة:

```
FROM Employee WHERE BirthDate > #01/01/1980#;*SELECT
FROM Employee WHERE BirthDate > '01/01/1980';*SELECT
FROM Employee WHERE BirthDate > "01/01/1980";*SELECT
```

ويمكن هنا أيضا استخدام أي من المعاملات المنطقية = أو < أو > أو <= أو >= أو <> . كما يمكن استخدام العبارات الشرطية السابقة.

دوال الإحصاء في SQL (SQL Aggregate Functions):

توفر SQL مجموعة من الدوال المضمنة في المترجم ولا تحتاج جهدا برمجيا لاستخدامها، فقط يكفي استدعاؤها لتعيد قيمة وحيدة (وليس جدولا).

- **لمعرفة عدد السجلات من أي جدول نستخدم الدالة COUNT، مثال:**

```
SELECT COUNT(*) FROM Employee;
```

لاحظ أنه لا ضرورة لتحديد حقل معين. يمكن أن نعطي المجموع اسما مستعارا، مثال:

```
SELECT COUNT(*) AS SumOfEmplyee FROM Employee;
```

- **تحسب الدالة AVG متوسط قيم حقل ما، مثال لحساب متوسط أعمال العمال:**

```
SELECT AVG(Age) FROM Employee;
```

- **تحسب الدوال MIN و MAX و SUM (على التوالي) أصغر وأكبر ومجموع قيم حقل محدد، مثال:**

```
SELECT SUM(EmpID), MIN(EmpID), MAX(EmpID) FROM Employee;
```

- **تستخدم عبارة GROUP BY غالبا مع دوال الإحصاء لتجميع سجلات الجدول وفق قيم حقل معين، مثال لمعرفة متوسط أعمار الطلبة حسب رقم القسم ClassNo:**

```
SELECT ClassNo, AVG(Age) FROM Student GROUP BY ClassNo;
```

- **إلى جانب ذلك هناك دوال مساعدة أخرى متوفرة في Access وSQL Server، نذكر كمثال TOP لتحديد عدد معين من أعلى الجدول من الحقول:**

```
SELECT TOP 10 * FROM Student
```

بنفس الطريقة لتحديد 50٪ من السجلات:

```
SELECT TOP 50 PERCENT * FROM Student
```

- **للاستعلام عن بيانات جدولين حيث تتساوى قيم حقولها يمكن استخدام عبارة كالآتي:**

```
SELECT Customer.*, Orders.* FROM Customer, Orders
WHERE Orders.CustNo = Customer.CustNo
```

- **لإضافة سجلات إلى الجداول نستخدم كلمة INSERT INTO، كالآتي:**

```
INSERT INTO Student(Name, Age) VALUES ('Amine', 10);
```

يمكن إضافة سجلات دون تحديد أسماء الحقول لكن مع مراعاة ترتيبها جميعا، مثال:

```
INSERT INTO Student VALUES ('Amine', 10);
```

- **لتعديل (تحديث) البيانات نستخدم عبارة UPDATE SET كالآتي:**

```
UPDATE Student SET Age = 12 WHERE Name = 'Amine';
```

ينبغي هنا إدراج عبارة شرط WHERE لتمييز السجل الذي ترغب في تعديله، وإلا فسيتم تغيير كافة السجلات في الجدول!

- لحذف سجل نستخدم كلمة DELETE مع عبارة شرط WHERE بنفس الطريقة:

```
DELETE FROM Student WHERE Name = 'Amine';
```

لحذف كافة سجلات الجدول Student:

```
DELETE FROM Student;
```

- ربط الجداول يعني عرض بيانات جدولين (أو أكثر) وفق شرط معين، ويتم باستخدام أحد عبارات الربط JOIN الآتية:

✓ الربط الداخلي INNER JOIN أو JOIN مباشرة. مثال:

```
SELECT * FROM Customer INNER JOIN Orders ON Customer.CustNo = Orders.CustNo
```

يتم إجراء هذا الربط لعرض جميع السجلات الموجودة في الجدولين معاً.

✓ الربط الخارجي من اليسار LEFT JOIN (أو LEFT OUTER JOIN في SQL Server)، مثال:

```
SELECT * FROM Customer LEFT JOIN Orders ON Customer.CustNo = Orders.CustNo
```

يتم إجراء هذا النوع من الربط لعرض جميع السجلات من كلا الجدولين وحتى التي ليس لها مقابل من الجدول اليسار.

✓ الربط الخارجي من اليمين RIGHT JOIN (أو RIGHT OUTER JOIN في SQL Server)، مثال:

```
SELECT * FROM Customer RIGHT JOIN Orders ON Customer.CustNo = Orders.CustNo
```

يتم إجراء هذا النوع من الربط لعرض جميع السجلات من كلا الجدولين وحتى التي ليس لها مقابل من الجدول اليمين.

✓ الربط الشامل FULL JOIN (أو FULL OUTER JOIN في SQL Server)، مثال:

```
SELECT * FROM Customer FULL JOIN Orders ON Customer.CustNo = Orders.CustNo
```

يعرض هذا النوع من الربط جميع السجلات من الجدولين بما فيها التي ليس لها مقابل من كلا الجدولين.

في SQL Server هناك أيضا الربط المتقاطع CROSS JOIN ويكتب مباشرة بهذا الشكل[2]:

```
SELECT * FROM Customer CROSS JOIN Orders
```

- عملية الاتحاد UNION: تسمح هذه العملية بضم جميع سجلات الجدولين:

```
SELECT * FROM Student
UNION
SELECT * FROM Person
```

لتتم عملية الاتحاد (UNION أو UNION ALL) يجب أن تحتوي الجداول نفس العدد من الحقول، وأن تكون الحقول من نفس نوع البيانات كما يجب أن تسرد الحقول وفق نفس الترتيب.

[2] للمزيد حول SQL Server انظر موضوع دورة تطبيقية لتصميم برنامج قواعد البيانات SQL Server في منتدى دلفي للعرب.

- عبارة HAVING: تستخدم غالبا مع دوال الإحصاء ضمن عبارة الشرط WHERE لأن هذا الأخير لا يمكن استخدامه مع دوال الإحصاء، وغالبا ما تسبقها عبارة التجميع GROUP BY، مثال[3]:

```
SELECT StudID ,Avg(Age) FROM Student
GROUP BY StudID
HAVING SUM(Age) < 20
```

تعليمات تعريف البيانات (DDL) Data Definition Language:

DDL تخص تعليمات التعامل مع هياكل قواعد البيانات، ومنها[4]:

- إنشاء قاعدة بيانات في SQL Server كالآتي:

```
CREATE DATABASE School;
```

- لحذف قاعدة بيانات في SQL Server نكتب:

```
DROP DATABASE School;
```

- إنشاء الجداول، مثال:

```
CREATE TABLE Student(Name CHAR(80), Age INT)
```

CHAR و INT هي أسماء أنواع المتغيرات وقد تختلف من RDBMS إلى آخر.

القيمة 80 تمثل طول الحقل Name

بتنفيذ هذه الجملة تم إنشاء جدول Student يحمل حقلين Name نصي وAge رقمي.

نضيف عبارة NOT NULL بعد اسم الحقل لجعله لا يقبل قيمة خالية، مثال:

```
CREATE TABLE Student(StudID INT NOT NULL, Name CHAR(80), Age INT)
```

هذه الأخيرة غير متوفرة في Paradox 7.

مثال لإنشاء حقل تعريفي للحقل ويأخذ قيم تصاعديا تلقائيا، في Paradox 7:

```
CREATE TABLE Student(StudID AUTOINC, Name CHAR(80), Age INT);
```

في Access:

```
CREATE TABLE Student(StudID AUTOINCREMENT, Name CHAR(80), Age INT);
```

في SQL Server:

```
CREATE TABLE Student(StudID INT IDENTITY, Name CHAR(80), Age INT);
```

فيما يخص Access وSQL Server هناك خيارات أخرى يمكن إضافتها (Options) إلى أي حقل، نذكر منها:

- UNIQUE لجعل الحقل فريد (قيمه لا تتكرر)، مثال:

```
CREATE TABLE Student(StudID UNIQUE, Name CHAR(80), Age INT);
```

- PRIMARY KEY: لجعل الحقل مفتاح أساسي، مثال:

```
CREATE TABLE Student(StudID PRIMARY KEY, Name CHAR(80), Age INT);
```

- FOREIGN KEY: لجعل الحقل مفتاح غريب من جدول آخر، مثال:

---

[3] هذا المثال لا يفيد في شيء، لكنه صحيح ويوضح الفكرة.

[4] هناك اختلافات صغيرة بين بعض معايير SQL، سأحاول استخدام الصيغة الأكثر توافقا لكل المعايير مع الإشارة أحيانا إليها.

```
CREATE TABLE Student(StudID INT PRIMARY KEY, Name CHAR(80), Age INT, PersonID
INT, CONSTRAINT FK_PersonID FOREIGN KEY (PersonID) REFERENCES
Person(PersonID));
```

وللحذف:

```
ALTER TABLE Student
DROP CONSTRAINT FK_PersonID
```

- CHECK: لإضافة قيد لفحص البيانات، مثلا الجملة الآتية تقوم بإنشاء جدول Student للطلبة مع منع إضافة طالب يفوق عمره 20 سنة:

```
CREATE TABLE Student(StudID INT PRIMARY KEY, Name CHAR(80), Age INT, CONSTRAINT
CHK_Age CHECK (Age < 20));
```

لإضافة قيد فحص البيانات بعد إنشاء الجدول:

```
ALTER TABLE Student ADD CONSTRAINT CHK_Person CHECK(Age > 30)
```

ولحذفه:

```
ALTER TABLE Student DROP CONSTRAINT CHK_Person
```

- DEFAULT: لإضافة قيمة افتراضية حالة عدم إدخال البيانات (Access فقط)، مثال:

```
CREATE TABLE Student(StudID INT, Name CHAR(80), Age INT DEFAULT 12);
```

في SQL Server:

```
ALTER TABLE Student
ADD CONSTRAINT DV_Age DEFAULT (12) FOR Age
```

ولحذفها في Access:

```
ALTER TABLE Student
ALTER COLUMN Age DROP DEFAULT
```

وفي SQL Server:

```
ALTER TABLE Student
DROP DV_Age
```

- حذف الجداول، مثال لحذف الجدول السابق:

```
DROP TABLE Student;
```

ينبغي تحديث الجداول إذا كان مرتبطا مع جداول أخرى كما يلي:

```
DROP TABLE Student CASCADE;
```

- إضافة حقول إلى الجدول، مثال إضافة الحقل Address إلى الجدول Student:

```
ALTER TABLE Student ADD COLUMN Address CHAR(255);
```

في SQL Server مثلا، تكتب هذه الجملة مباشرة كالآتي:

```
ALTER TABLE Student ADD Address CHAR(255);
```

- لحذف أحد الحقول نكتب:

```
ALTER TABLE Student DROP COLUMN Address;
```

- لإنشاء فهرس (Index) نكتب:

```
CREATE INDEX StudIDX ON Student(StudID);
```

يمكن طبعا إضافة عدة حقول إلى الفهرس بهذا الشكل:

```
CREATE INDEX StudIDX ON Student(StudID, Age);
```

- ونكتب لحذف الفهرس:

```
DROP INDEX Student.StudIDX
```

في Access و SQL Server نكتب:

```
DROP INDEX StudIDX ON Student
```

- تعليمة SELECT INTO: تقوم بإضافة مجموعة من السجلات المحددة من جدول إلى جدول آخر، تستخدم هذه التعليمة غالبا لحفظ نسخة احتياطية (Backup)، مثال:

```
SELECT * INTO Student_Backup FROM Student
```

أو لحفظ نسخة احتياطية في ملف خارجي (Access فقط) ينبغي إنشاؤه مسبقا:

```
SELECT * INTO Student_Backup IN 'Backup.mdb' FROM Student
```

ملاحظة:

- **تعليمة** SELECT INTO **غير متوفرة في** Paradox
- **يمكن إضافة شروط** (WHERE) **أو استخدام استعلام ربط** (JOIN).

- **القيمة المعدومة** NULL: **نقول عن السجلات التي يقم المستخدم بإدخال بيانات إليها بأنها** NULL، **ولا يعني ذلك أنها تحمل القيمة** 0 (**إذا كانت عددية) ولا القيمة** " (**إذا كانت نصية)، وإنما تعامل معاملة خاصة من قبل محلل** SQL.

لفحص قيمة حقل من الحقول إذا كانت NULL **أم لا يمكن أن نكتب**:

```
SELECT * From Student WHERE Age IS NOT NULL
```

باختصار هذه مجمل تعليمات جمل SQL **الأكثر استعمالا وتوافقا مع معظم معايير** SQL.

ما تم سرده من الأمثلة متوافق مع Paradox، Access، SQL Server **مع بعض الاختلافات المذكورة، وليس بالضرورة أن يتوافق مع المعايير الموافقة لها** (SQL-92، SQL-89، SQL-99).
مجمل نظم قواعد البيانات الشهيرة لا يخرج عن تطبيق أحد هذه المعايير مع تعديلات صغيرة، فمثلا المعيار المستخدم في MySQL **لا يكاد يختلف عن المعيار** SQL-92، **و**Firebird **يدعم معيار** SQL-92 **بشكل كامل ومعظم صيغ** SQL-99.
والله أعلم.

المراجع:

http://www.w3schools.com
http://www.e-naxos.com
http://www.thedbcommunity.com

استخدام الكائنات (Objects)

من الملاحظ مِن قبل الجميع قِلة استخدام الـ Objects في كتابة الدوال و الإجراءات, رغماً عن ظهورها منذ فترة طويلة, حيث يعتبر TObject الوحدة الأساسية المسؤولة عن جميع الكائنات و مكوناتها, في البداية تـُـنشأ الكائنات في الذاكرة ثم تستخدم ثم تحرر بعد الانتهاء من استخدامها.

يُكتب الكائن ضمن التعليمة type و من ثم يعرف في قسم المتغيرات.
جميع الكائنات لها تعليمة إنشاء Create و إجراء تحرير Free أما البقية فلا تستخدم بقدر استخدامها.
مثال على تعريف كائن (تضاف هذه الأسطر للمشروع تحت قسم : interface

```
interface

uses Windows;

type
  TMyObject = class(TObject)
  public
    // يعرف هنا كل من
    // function, procedure, property, ..
  end;
```

و الاستدعاء :

```
var
  MyObject: TMyObject;
Begin
  MyObject := TMyObject.Create;
  Try
    // يمكننا الأن استخدام دوال او اجرائات او باقي الأشياء من الكائن
  finally
    MyObject.Free;
  end;
end;
```

سؤال: لماذا وضعنا دوال الكائن بين الـ Try و Finally ؟

جواب: لتأكد بأن الكائن سوف يحرر من الذاكرة حتى وإن صادفت العملية أخطاء.

سؤال: ما الفائدة من استعمال الكائنات ؟

أجوبة:

1. **ترتيب المشروع و سهولة فهمه (ففي البرامج الضخمة تستعمل بكثرة نضراً لترتيبها و سهولة تتبع الدالة أو الأجراء).**
2. **تستخدم في إنشاء المكونات.**
3. **تستخدم في اختصار الكود, وذلك عن طريق الـ Property (حيث أن أغلب دوال الـ VCL مكتوبة بهذه الطريقة).**

استخدام Property في كتابة إجراء للقارئة و الكتابة :

الـ Property خاصية تتيح لنا إنشاء دالة للقراءة أو إجراء للكتبة أو دالة و إجراء للقراءة و الكتابة.

ربما قد لا يعلم المبتدئ ما هي القراءة أو الكتابة (هذا توضيح بسيط):

مثلاً لدينا مكون Edit نريد أن نقرأ منه خاصية معينة, مثلاً الـ Enabled لنرى هل هو مفعل أو لا :

```
If Edit1.Enabled then // .....
```
قرائة الخاصية property

أما الكتابة:

```
Edit1.Enabled := Not Edit1.Enabled;
```
او
```
Edit1.Enabled := True;
```
كتابة الخاصية property

كتابة الخاصية Property

إذا نريد كتابة خاصية سوف نستخدمها في مشروع الـ API تقوم بالقراءة و الكتابة في خاصية الـ Checked لـ CheckBox

بعد إضافة المكون إلى المشروع بواسطة محرر الريسورس, نفرض أن الـ ID (المعرف) الخاص بها هو 1001.
نبدأ بإنشاء وحدة جديدة Unit (File > New > Unit) تحت اسم uCode.pas نبدأ بالكتابة بها

```
unit uCode;

interface

uses Windows;

type
  TMyCheckBox = class(TObject)
  public
    Property Checked: Boolean read GetCheck write SetChecked;
  end;


implementation

end.
```

عينا TMyCheckBox ككائن TObject ...
كتبنا الـخاصية الخاصة بنا Checked ثم نضغط على CTRL + SHIFT + C ليقوم الـ IDE الخاص بدلفي بإنشاء دالة للقراءة و إجراء للكتابة تحت قسم implementation.

```
implementation

{ TMyCheckBox }

function TMyCheckBox.GetCheck: Boolean;
begin

end;

procedure TMyCheckBox.SetChecked(const Value: Boolean);
begin

end;
```

طبعاً كتابة الكود الصحيح مرهون بمعرفتك بدوال الـ API...

بعد الضغط على CTRL + SHIFT + C سيقوم الـ IDE أيضاً بالتالي المبين باللون الأحمر,
نضيف إليه متغيرين للحصول على عنوان النافذة و الـ ID للمكون (المبين باللون الأخضر) :

```
type
  TMyCheckBox = class(TObject)
  private
    WinHandle, ItemID: HWND;
    function GetCheck: Boolean;
    procedure SetChecked(const Value: Boolean);
  public
    Property Checked: Boolean read GetCheck write SetChecked;
  end;
```

سبب كتابتنا للمتغيرات و سبب قيام الـ IDE بوضع الدالة و الأجراء تحت قسم Private هو لعدم ظهورها في باقي الوحدات Unit او في المشروع, فقط تستخدم من قبل الوحدة ذاتها.
لنفرض أننا قمنا بتحويلها جميعها إلى قسم Public ماذا سيحدث ؟؟
ببساطة عملية استدعائها ستكون مفعلة من قبل باقي الوحدات و هذا الذي لا نريده لأننا لا نستفيد منه, الصورة أدناه تبين إذا قمنا بالعملية :

var	WinHandle : HWND;
var	ItemId : HWND;
function	GetCheck : Boolean;
procedure	SetChecked(const Value: Boolean);
property	Checked : Boolean;
constructor	Create;
procedure	Free;
function	InitInstance(Instance: Pointer): TObject;
procedure	CleanupInstance;
function	ClassType: TClass;
function	ClassName: ShortString;
function	ClassNameIs(const Name: String): Boolean;
function	ClassParent: TClass;
function	ClassInfo: Pointer;

لذلك سوف نبقي على الدالة و الأجراء في قسم Private لكن المتغيرات WinHandle و ItemId سوف نحولها إلى قسم Public حتى نستطيع تمرير مقبض النافذة و الـ ID الخاص بالمكون.

كتابة الكود و استدعائه :

أولاً نضيف ملف Messgaes إلى الـ Uses.

نكتب في الدالة function :

```
function TMyCheckBox.GetCheck: Boolean;
begin
  Result := Boolean(IsDlgButtonChecked(WinHandle, GetDlgItem(WinHandle,
ItemId)));
end;
```

يعلم البعض أن معظم دوال النظام ترجع قيمة 1 في حالة True و 0 أو أي رقم آخر في حالة خطأ (و هذه الدالة احدها) لذلك سبقت الأمر بـ Boolean ليحول لنا الوحدة إلى قيم صواب و خطأ, ليقربها و يجعلها شبيهة بدوال الـ VCL (للمزيد عن عملية تحويل الوحدات راجع موقع دلفي للعرب ..)

أما في الأجراء فيكون كالتالي :

```
procedure TMyCheckBox.SetChecked(const Value: Boolean);
begin
  SendDlgItemMessage(WinHandle, ItemId, BM_SETCHECK, Integer(Value), 0);
end;
```

استدعاء الأوامر:

نقوم بتعرف المتغير في الجزء الأعلى للشروع :

```
{$R Res.res}

var
  MyCheckBox1: TMyCheckBox;
```

نقوم باستدعاء الكائن في بداية تشغيل البرنامج و التأكد بأنه سوف يتحرر مع غلق البرنامج:

```
begin
  MyCheckBox1 := TMyCheckBox.Create;
  Try
    DialogBoxParam(HInstance, PChar(1000), 0, @Main, 0);
  finally
    MyCheckBox1.Free;
    ExitProcess(hInstance);
  end;
end.
```

سؤال : لماذا لم نقم بتحريرهِ في عند غلق البرنامج WM_CLOSE (WM_CLOSE شبيه لـ OnClose في الـ VCL)

جواب: لأنه دالة DialogBoxParam تنفذ الى ان يتم استدعاء دالة EndDialog لأنهاء الدالة ثم تنفيذ الأسطر السفلية (للتأكد ضع نقطة توقف ستلاحظ أن الدالة ستنفذ إلى أن تخرج من الدالة ثم تستدعي أمر التحرير.

سؤال: ماذا سيحدث إذا لم تكن الدالة DialogBoxParam تتوقف للتنفيذ إلى وقت استدعاء الدالة EndDialog ؟

جواب: ببساطة ستحرر الكائن و يحدث أخطاء, في هذهِ الحالة يجب ا ن نحررها في حدث WM_CLOSE.

الآن نقوم بتمرير مقبض النافذة و الـ ID الخاص بالمكون في حدث WM_INITDIALOG (الذي يقابل حدث OnCreate في الـ VCL)

```
WM_INITDIALOG:
  Begin
    MyCheckBox1.WinHandle := Win;
    MyCheckBox1.ItemId := 1001;
  end;
```

علماً بأن Win هو مقبض النافذة و 1001 هو الـ ID (المعرف) الخاص بالمكون الذي حددناه بواسطة محرر الريسورس عند إضافة المكون.

و الآن نكتب في حدث النقر على الزر أمر القراءة أو الكتابة بكل سهولة:

قراءة :

```
if MyCheckBox1.Checked then
  MessageBox(Win, 'Checked !', '', $40)
else
  MessageBox(Win, 'UnChecked !', '', $40);
```

كتابة :

```
MyCheckBox1.Checked := Not MyCheckBox1.Checked;
```

أو

```
MyCheckBox1.Checked  := True;
```

تقنية من تقنيات رسم الدوال

لكي نقوم بعملية الرسم في دلفي نحتاج لمكون يحتوى على خاصية Canvas وهى في حد ذاتها عبارة عن Class تحتوى على عدة خصائص و دوال تساعدنا في عملية الرسم، نختار مكون مثل Image المخصص لذلك.

من بين المشاكل التي سوف تواجهنا:

اختلاف موقع الإحداثيات:

مثلا، الإحداثية (0 ،0) بالنسبة للمكون Image تتوقع في اعلي يسار هذا المكون على عكس طريقة الرسم التقليدية أي الرسم عن طريق المعالم ... لذالك نحتاج إلى عملية تحويل موقع هذه الإحداثية.

طريقة حساب الإحداثيات : F(X) = Y

هل نحسبها بالطريقة التقليدية أي نختار مجال معين يحتوى على أعداد سالبة وموجبة يتم تعويضها في الدالة لنحصل على الإحداثيات الواجب ربطها ؟

الجواب: لا

لأنه ربما تكون هذه الإحداثيات متباعدة و بالتالي الرسم يكون غير دقيق لأننا سنعتمد على الدالتين

Canvas.lineTo و Canvas.MoveTo

لذلك وجب علينا إيجاد عملية معينة تضمن أن كل الإحداثيات الناتجة تكون متقاربة.

تحديد أبعاد المعلم: تحديد وحدة القياس

مثلا: X1 = 26 Pixel

إذا لم نعر هذا الجانب اهتماما فان شكل رسم الدالة سيكون صغير جدا و غير واضح لان أبعاد المعلم ستساوي:

X 1 = 1 Pixel

و عادتا ما يرمز لهذه العملية بــ DX

خطوات الوصول إلى الحل

لنفرض انه امامنا دالة من نوع **كثير حدود من الدرجة الثانية** على الشكل التالي: Ax² + Bx + C

فى حلنا هذا نحتاج الى **دالتين** فى **دلفي** فقط

الدالة الأولى ـ لحساب قيمة : Y

و دالك عن طريق التمرير لها كل من **معلمات النمودج** و قيمة المتغير X وهى تقوم بحساب Y على أساس شكل الدالة.

```
function F_de_X(A, B, C, X: Real): Real;
begin
  Result := (
             Sqr(X) * (A) +
                X   * (B) +
                      (C)
           );
end;
```

الدالة الثانية:

وضيفتها الأساسية هي رسم الدالة ... لكن قبل هذا يجب تحضير بعض العمليات:

نبدأ بمعالجة الثوابت:

هناك 3 أعداد ثابتة لا تتغير و هي (200 | 0.1 | ـ10)

200: تعبر عن عدد **الحلقات** التي نحتاجها لعملية الرسم ... وعدد الحلقات لا يتغير مهما تغير **ارتفاع** أو **عرض** الصورة التي نرسم عليه.

و بالنسبة **لارتفاع** و **عرض** الصورة فهو ليس اختياري بتمام معنى الكلمة بل هو نصف اختياري (إن صحة التعبير) لأنه يكون دائما على أساس مقياس معين وهو (**الارتفاع = نصف العرض**) فمثلا إذا أردنا أن يكون العرض 700 فان الارتفاع يستخرج بعلاقة ثلاثية.

```
001[Pixels, Width]  -------------> 0.5[Pixels, Height]
700[Pixels, Width]  -------------> ? [Pixels, Height]
```

يمكنك البحث عن طريقة أخرى لا تخضع لقيد (الارتفاع = نصف العرض)

0.1: يعبر عن قيمة ثابتة تضاف إلى X بعدد تعاقب الحلقات.

10- : هي القيمة الأولى المخزنة في X.

أشبه بثابت DX:

هى عبارة عن قيمة تعبر عن إبعاد المعلم أو وحدت القياس وهى ثابت أثناء عملية الرسم أي لا تتغير أثناء عملية حساب الإحداثيات و رسمها.

عند إعطاء DX القيمة 10 فان وحدة القياس تكون تقريبا (X1 = 26 pixel)، هذه القيمة عبارة عن قيمة اختيارية، يمكن الاستعانة بأي قيمة أخرى.

حساب الإحداثيات التي تستعمل في عملية الرسم:

يتم حسابها اعتمادا على الإحداثيات المحسوبة مسبقا (X , F(X))

و طول و عرض الصورة كما يلي:

```
 How To Calculate the coordinates
====================================
  X  = (X_a1 / X_a2) * Width .
====================================
      X_a1 = (100% DX) + _X
      X_a2 = (200% DX)


====================================
  Y  = (Y_a1 / Y_a2) * Height .
====================================
      Y_a1 = (050% DX) - _Y
      Y_a2 = (100% DX)


====================================
     Canvas.lineTo( X ,  Y )
====================================
```

شكل الإجراء

```
procedure _Draw(
                  Canvas: TCanvas;

                  A, B, C: Real;

                  Width, Height: Integer;

                  DX: Real;

                  PenColor: TColor
                );
var
  I: 1..200;
  X, Y: Real;
  StartPoint: Boolean;
begin
  Canvas.Pen.Color := PenColor;

  X := -10;

  for I := Low(I) to High(I) do
  begin

    X := X + 0.1;

    begin
      Y := F_de_X(A, B, C, X);

      if not(StartPoint) then  // Starting Points
        Canvas.MoveTo(
                      Round( (   ((001 * DX) + X) / (2 * DX)    ) *  Width   ),
                      Round( (   ((0.5 * DX) - Y) / (1 * DX)    ) *  Height  )
                     )
      else
        Canvas.LineTo(
                      Round( (   ((001 * DX) + X) / (2 * DX)    ) *  Width   ),
                      Round( (   ((0.5 * DX) - Y) / (1 * DX)    ) *  Height  )
                     );
      StartPoint := True;
    end;
  end;
end;
```

هذا الإجراء صالح بالنسبة لدوال كثيرة الحدود و كل الدوال التي تقبل كل الأعداد أما الدوال التي تقبل مجال معين فيجب إجراء تغيير بسيط أو بالأحرى شرط صغير فقط على حسب الدالة التي تتعامل معها

مثال: كثير حدود من الدرجة الثانية

$$X^2 - 4X + 3$$

```
procedure(Canvas: TCanvas; A: Real; B: Real; C: Real; Width: Integer; Height: Integer; DX: Real; PenColor: TColor)
```

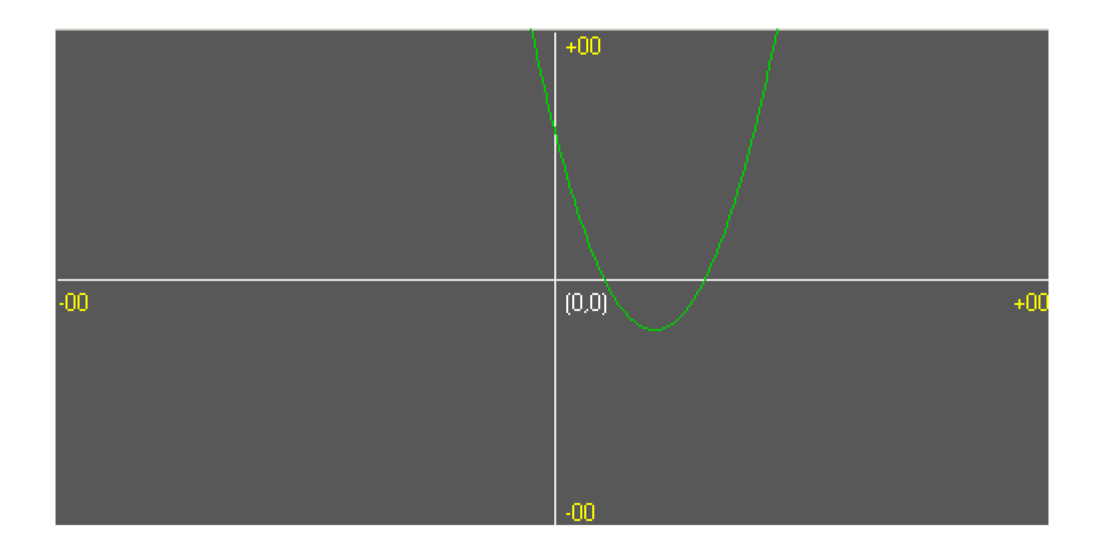
```
_Draw(
        img1.Canvas,

        +1,   // a
        -4,   // b
        +3,   // c

        img1.Width,
        img1.Height,

        10,

        clRed
      );
```



شكل الإجراء بالنسبة للدوال التي تقبل مجال معين

لنفرض انه أمامنا دالة لوغرتمية من الشكل التالي:

$Ln(X^2 - 0X + 1)$

إجراء حساب : Y

```
function F_de_X(A, B, C, X: Real): Real;
begin
  Result := Ln(
               Sqr(X) * (A) +
                   X  * (B) +
                        (C)
             );
end;
```

إجراء الرسم مع التعديلات الازمة

```
begin
  Canvas.Pen.Color := PenColor;

  X := -10;

  for I := Low(I) to High(I) do
  begin

    X := X + 0.1;

    if (
          Sqr(X) * (A) +
              X    (B) +
                   (C)
       ) > 0
    then

    begin
      Y := F_de_X(A, B, C, X);

      if not(StartPoint) then  // Starting Points
        Canvas.MoveTo(
                      Round( (   ((001 * DX) + X) / (2 * DX)    ) *  Width   ),
                      Round( (   ((0.5 * DX) - Y) / (1 * DX)    ) *  Height  )
                      )
      else
        Canvas.LineTo(
                      Round( (   ((001 * DX) + X) / (2 * DX)    ) *  Width   ),
                      Round( (   ((0.5 * DX) - Y) / (1 * DX)    ) *  Height  )
                      );
      StartPoint := True;
    end;
  end;
end;
```

القسم المحدد بالبرتقالي هو مخصص لمراقبة الأعداد المررة للدالة المتعامل معها فان كان هذا العدد ينتمي للمجال الذي تقبله هذه الدالة سيبدأ في عملية الرسم.

ملاحظة: بالنسبة لإبعاد الصورة التي نرسم عليها يجب إن يكون العرض = ضعف الارتفاع

نظرة على مكون ComboBox

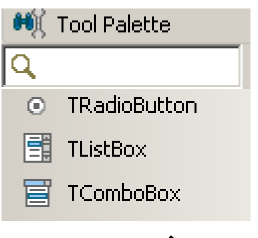
كما جرت العادة في إصدارات مجلة دلفي للعرب، نخصص في كل مرة موضوع تعريفي بسيط عن مكون من مكونات دلفي، طبعا على المتتبع أن يقوم بالبحث و الخوض أكثر في خصائص المكون التي لو يتم التطرق إليها.

ما هو المكون:

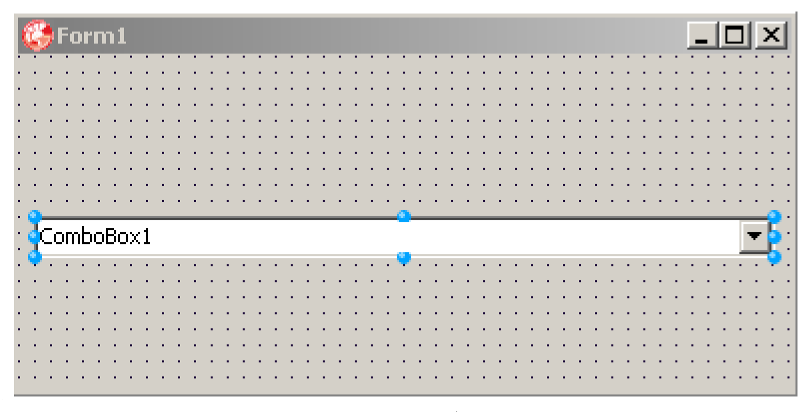
المكون موضوع المقالة هو عبارة عن حقل يقبل الكتابة في شكل طولي مثل مكون TEdit مع إعطاء إمكانية معالجة القوائم سواء في مرحلة التصميم أو التنفيذ.

كيف التعامل مع المكون؟:

نذهب إلى تبويب Standard ثم نختار في القائمة مكون ComboBox – شكل 01

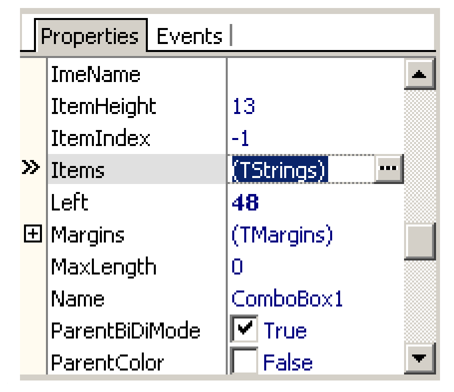


شكل 01



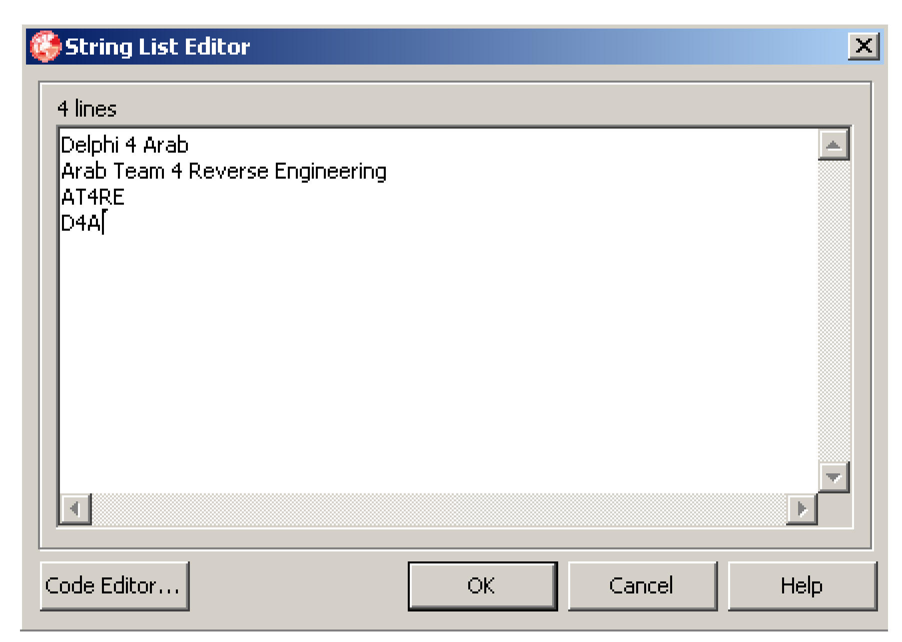
شكل 02

ننشئ مشروع جديد و نضيف المكون للمشروع – شكل 02



شكل 03

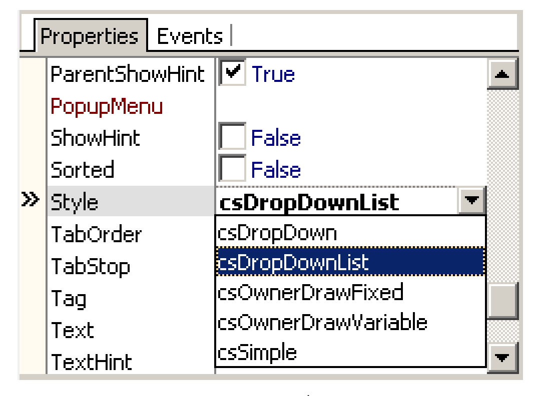
لإضافة قائمة يحفظها المكون، نقوم بتحديد المكون في واجهة المشروع ثم نذهب إلي: Properties **ثم** Items **ثم نختار** TStrings – **شكل** 03



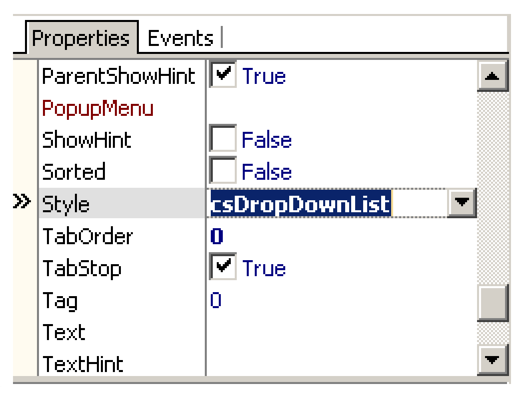
شكل 04

في المحرر الذي سوف يظهر لنا عند اختيار TStrings **نقوم بإدخال القائمة مع مراعاة عدم ترك فراغات في أول أو بين أو آخر القائمة. شكل** 04

المكون يحتوي على أنماط Styles من بينها نختار النمط المستعمل بكثرة : csDropDownList انظر الشكل 05 و الشكل06.



الشكل 05



الشكل 06

Properties	Events
Hint	
ImeMode	imDontCare
ImeName	
ItemHeight	13
ItemIndex	-1
Items	(TStrings)
Left	8
⊞ Margins	(TMargins)
MaxLength	0
Name	ComboBox1

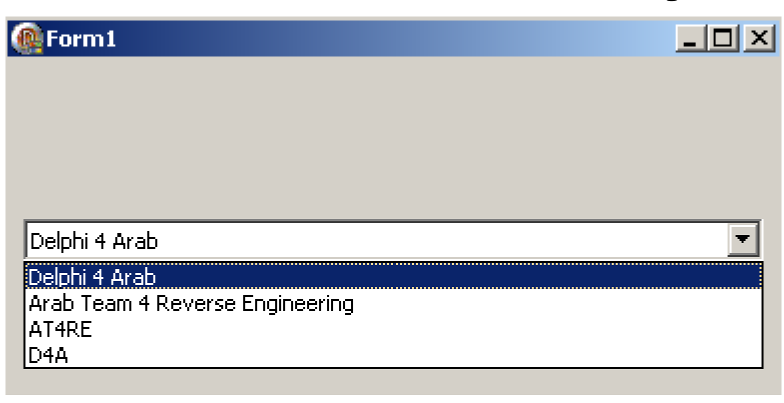
شكل 07

هام: إذا تركنا قيمة ItemIndex تساوي (ناقص واحد) 1- فان المكون لن يظهر أول عنصر في القائمة، لذا من الأحسن تغيير قيمة ItemIndex إلى (صفر) – شكل 07 و شكل 08



شكل 08

نجرب المكون بعد تشغيل المشروع – شكل 09



شكل 09

من الشائع و المستحسن أن نقوم بالتعامل مع العناصر المحفوظة في المكون برمجيا عن طريق معالجة ترتيبها اعتمادا على ItemIndex.

لاحظ المثال العملي التالي: نضيف كل من Button, Label **و** ComboBox **لمشروع جديد.**

نختار Case of **للقيام بالعملية بسلاسة.**

في حدث النقر الخاص بمكون Button **نكتب الأوامر التالية: ـ شكل** 10

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  case ComboBox1.ItemIndex of
    0:
      Label1.Caption := ComboBox1.Items.Strings[0];
    1:
      Label1.Caption := ComboBox1.Items.Strings[1];
    2:
      Label1.Caption := ComboBox1.Items.Strings[2];
    3:
      Label1.Caption := ComboBox1.Items.Strings[3]
    else
      Label1.Caption := '';
  end;
end;

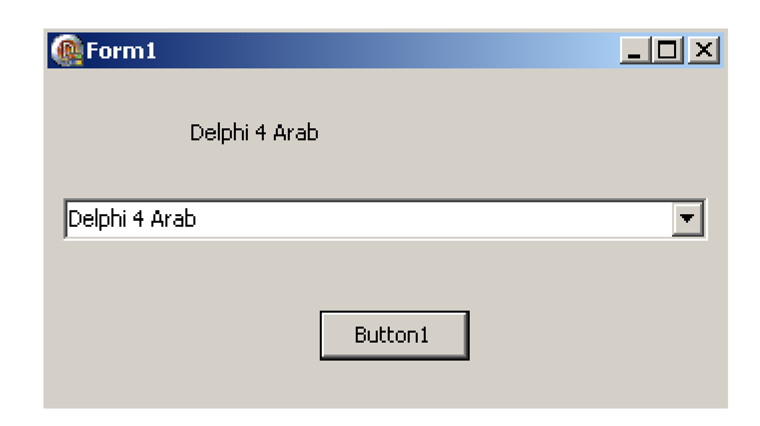
end.
```

شكل 10

<u>فكرة عامة للمثال العملي:</u>

Case **سوف تبدأ بقراءة القيمة الموجودة في** ItemIndex **و تقوم بتنفيذ الأمر المناسب حسب القيم الثابتة المعطاة من** 0 **إلى** 3 **و في حالة قراءة قيمة خارج ما تم وضعه كثابت، تنفذ الأمر الخاص بــ** else

مكون Label1 **يأخذ القيمة النصية الموجودة في** Strings **التابعة للــ** Items **حسب ما تم إدخاله من قيمة الترتيب الثابت.**



تنفيذ المشروع و تجربته شكل 11

برامج لها علاقة بدلفي

Meerkat - Advanced kernel mode driver GUI for KmdKit4D (LanHua (mickeylan))

نبذة عن الأداة:

Meerkat **هي عبارة عن أداة قيد التطوير من برمجة** STRELiTZIA **لتسهيل بناء ملفات نظام تعمل على مستوى الكرنل** Kernel mode، **تحتوي على عدة خصائص تسمح لنا بنشاء ملفات** Drivers ـ sys.* **بكل سهولة مع إعطاء إمكانية معالجة تشغيل الدرايفر و إظهار رسائل مراحل البناء و التشغيل.**

Meerkat Kernel mode driver Kit 4 Delphi | Advanced GUI

Build | Options | Driver manager | Informations

Index	Report	Status
Meerkat	Advanced GUI for KmdKit4D | by STRELiTZIA | AT4RE	
KmdKit4D	Kernel mode driver kit for Delphi	
1	Embarcadero Delphi for Win32 compiler version 21.0	COMPILE
2	Copyright (c) 1983,2009 Embarcadero Technologies, Inc.	
3	BasicDriver.pas(1)	
4	BasicDriver.pas(1)	
5	BasicDriver.pas(1)	
6	BasicDriver.pas(5)	
7	BasicDriver.pas(5)	
8	BasicDriver.pas(5)	
9	BasicDriver.pas(5)	
10	BasicDriver.pas(5)	
11	BasicDriver.pas(7)	
12	BasicDriver.pas(23)	
13	24 lines, 0.45 seconds, 99 bytes code, 4 bytes data.	
14	RMCOFF 2.0.0.87 Build on 2010.01.01 16:00:36	RMCOFF
15	OMF2COFF utility toolkit for KmdKit4D(Delphi & BCB)	
16	Copyright (C) 2008-2009 by LanHua(mickeylan[RCT])	
17	Process Ok...	
18	Microsoft (R) Incremental Linker Version 7.00.9210	LINK
19	Copyright (C) Microsoft Corporation. All rights reserved.	
	Out: C:\release\BasicDriver.sys	RELEASE
	Operation terminate at: 16:12:15 | 05/08/2010	

واجهة البناء

Meerkat Kernel mode driver Kit 4 Delphi | Advanced GUI KmdKit4D

Build | Options | Driver manager | Informations

Compiler options

- [x] A= Set unit alias
- [x] B = Build all units
- [ ] CG = GUI target
- [x] JP = Generate .obj file
- [x] U = Unit directories
- [ ] C- = (Not) Evaluate assertions at runtime
- [ ] D- = (Not) Debug information
- [ ] G- = (Not) Use imported data references
- [x] H- = (Not) Use long strings by default
- [x] I- = (Not) I/O checking
- [ ] L- = (Not) Local debug symbols
- [ ] P- = (Not) Open string params
- [x] O+ = Optimization
- [ ] V- = (Not) Strict var-strings
- [ ] W+ = Generate stack frames
- [ ] Y- = (Not) Symbol reference info

Link options

- [ ] FORCE UNRESOLVED (Not Recommended)
- [ ] FORCE MULTIPLE (Not Recommended)
- [ ] NOLOGO (do not display linker Copyright)
- [x] ALIGN: 32
- [x] IMAGE BASE: 0x10000
- [x] SUBSYSTEM: NATIVE
- [x] DRIVER
- [x] DRIVER ENTRY: DriverEntry
- [ ] IMAGE VERSION: 1.0
- [ ] DEBUG (Add Debug info)

Build options

- [ ] Create MakeFile / Batch
- [x] Create cleaner

Sources

Rebuild sources (Update units)

Maker: Version 5.4 Copyright (c) 1987, 2009 CodeGear

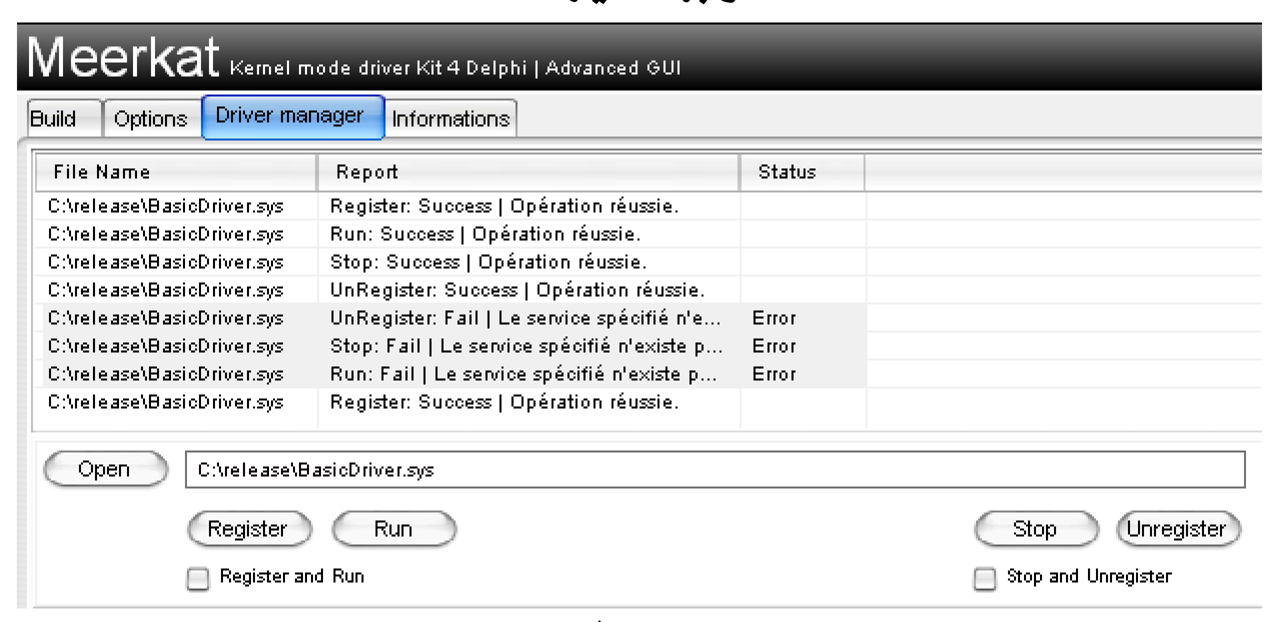
Compiler : Embarcadero Delphi for WIN32 Compiler version 21
Copyright (c) 1983,2009 Embarcadero Technologies, Inc.

COFF Converter : RMCOFF 2.0 COFF Utility tool for KmdKit4D (Delphi & BCB)
LanHua (mickeylan)

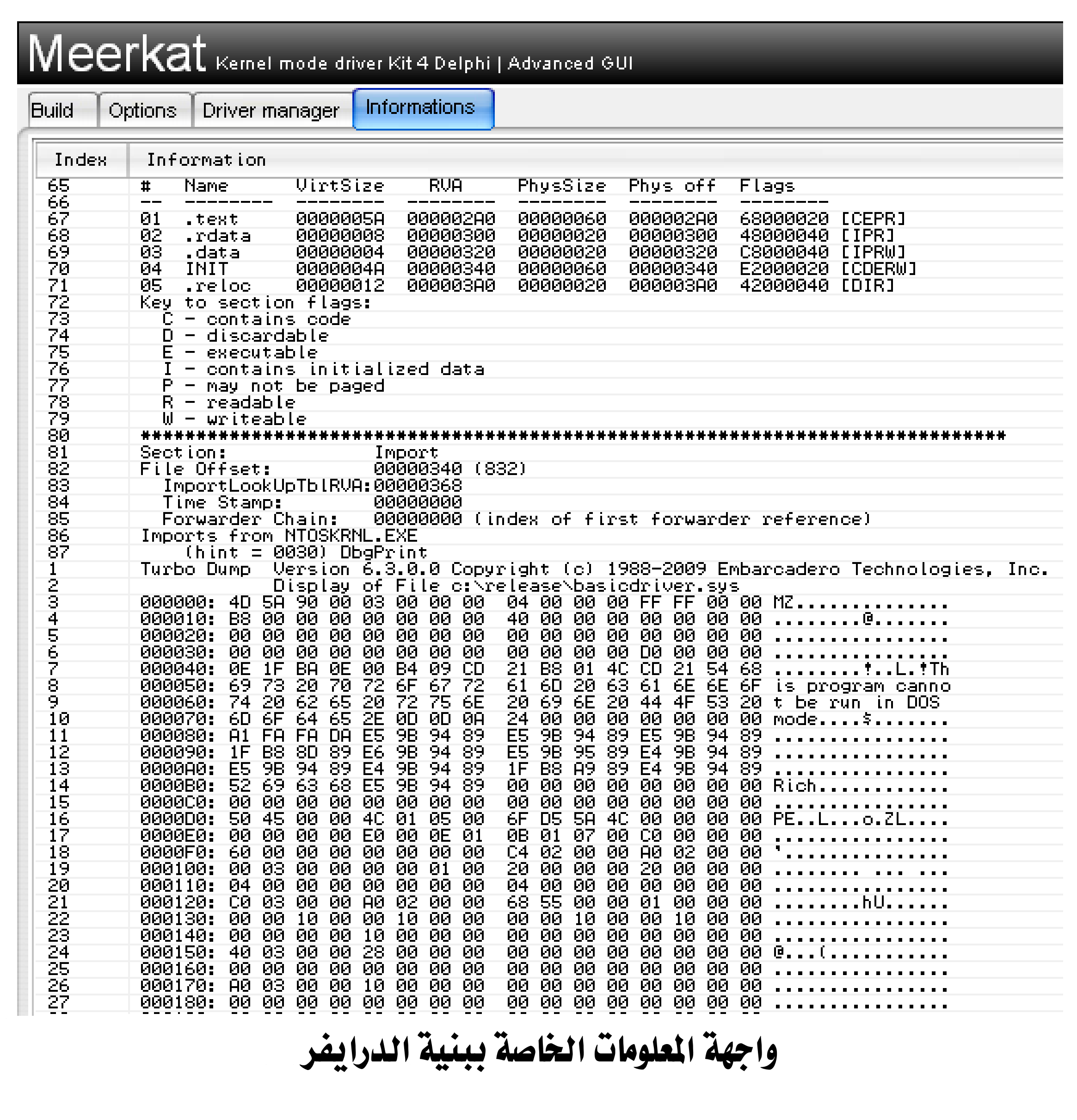
Linker: Microsoft (R) Incremental Linker Version 7.00.9210

STRELiTZIA | AT4RE

واجهة الخيارات



واجهة التشغيل



واجهة المعلومات الخاصة ببنية الدرايفر

ترقبوا إصدار جديد للأداة أول أيام رمضان إن شاء الله

رابط التحميل:

www.delphi4arab.com

حل تمرين العدد 02 – بقلم TF6M

حل التمرين كان بمشاركة وحيدة من عضو من المنتدى بصيغة أوامر مع إرفاق الملفات المصدرية، يمكنكم تجربته و طرح أي غموض في قسم الأسئلة في منتدى دلفي للعرب.

```
{********************************************* Coded By TF6M}
                        //15-07-2010
                        //Delphi4Arab
{********************************************* Coded By TF6M}
unit SmplUnit;
{$WARNINGS OFF}
{$HINTS OFF}
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ShellApi;

type
  TMainFrm = class(TForm)
    BackGnd: TShape;
    Label1: TLabel;
    iMemo: TMemo;
    tLabel: TLabel;
    Label2: TLabel;
    dSLabel: TLabel;
    sForm: TCheckBox;
    procedure BackGndMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Label2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure sFormClick(Sender: TObject);
  private
   Procedure nHDevice(Var M:TMessage); message WM_DEVICECHANGE;
   Function GetNewDriveName : String;
   Procedure GetDrivInfo;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  MainFrm: TMainFrm;

implementation

{$R *.dfm}
```

```
procedure TMainFrm.BackGndMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 ReleaseCapture;
 SendMessage(Handle,Wm_SysCommand,$F012,0);
end;

Function TMainFrm.GetNewDriveName : String;
Var
 _i : Byte;
Begin
Result := '';
for _i := $61 To $7A Do
Begin
If GetDriveType(PChar(Chr(_i) +':\')) = DRIVE_REMOVABLE Then
 Begin
  Result := Chr(_i) + ':\';
  dSLabel.Caption:= ConCat('dStat : ('
   ,Chr(_i)
   ,') Disque Amovible');
 End {Else
  MessageBoxA(0,Pchar(Format('Exit Code : %X',[GetLastError]))
               ,Nil,$2000 + $40);}
End;
End;

procedure TMainFrm.GetDrivInfo;
Var
 BuffNom : Array[0..255]of Char;
 BuffSys : Array[0..255]of Char;
 Serie, Long, Flags : DWord;
 FreeBytesAvailable, TotalNumberofBytes, TotalNumberofFreeBytes : Int64;
begin
 If GetVolumeInformation(Pchar(GetNewDriveName),BuffNom, SizeOf(BuffNom)
                      ,@Serie,Long,Flags
                      ,BuffSys, SizeOf(BuffSys)) Then
 Begin
 GetDiskFreeSpaceEx(Pchar(GetNewDriveName),FreeBytesAvailable
                    ,TotalNumberofBytes
                    ,@TotalNumberofFreeBytes);
 With iMemo.Lines Do
  Begin
  Add('');
  Add('USB Name : ' + BuffNom);
  Add('File System : '+ BuffSys);
  Add('Serial Number : $'+ IntToHex(Serie,8));
  Add('Free  : ' + IntToStr(TotalNumberofFreeBytes)+ ' Bytes');
  Add('Total : ' + IntToStr(TotalNumberOfBytes)+ ' Bytes');
  End;
```

```
 If MessageBoxA(0,Pchar('Open It ?')
               ,Nil,$2000 + $40 + $01) = $01 Then
  ShellExecute(0,Nil,Pchar(GetNewDriveName),Nil,Nil,1);
 End Else
  Begin
  iMemo.Lines.Add('Nothing detected!');
  dSLabel.Caption:= 'dStat : (.) Disque Amovible';
  End;
 // MessageBoxA(0,Pchar(Format('Exit Code : %X',[GetLastError]))
               //,Nil,$2000 + $40);
end;

procedure TMainFrm.Label2Click(Sender: TObject);
begin
 ExitProcess(GetLastError);
end;

procedure TMainFrm.nHDevice(var M: TMessage);
begin
Case M.wParam Of
 $8004{DBT_DEVICEREMOVECOMPLETE}:
   Begin
    iMemo.Clear;
    iMemo.Lines.Add('A device or piece of media has been removed ...');
    iMemo.Lines.Add('Action Time : ' + TimeToStr(Now));
    dSLabel.Caption:= 'dStat : (.) Disque Amovible';
   End;
 $8000{DBT_DEVICEARRIVAL}:
   Begin
    iMemo.Clear;
    iMemo.Lines.Add('A device or piece of media has been inserted and is now
available...');
    iMemo.Lines.Add('Action Time : ' + TimeToStr(Now));;
    GetDrivInfo;
   End;
 {Else
 If M.wParam <> 0 Then
  MessageBoxA(0,Pchar(Format('Exit Code : %X',[M.wParam]))
               ,Nil,$2000 + $40);
 http://msdn.microsoft.com/en-us/library/aa363480%28VS.85%29.aspx}

 End;
end;

procedure TMainFrm.FormActivate(Sender: TObject);
begin
 GetDrivInfo;
end;
```

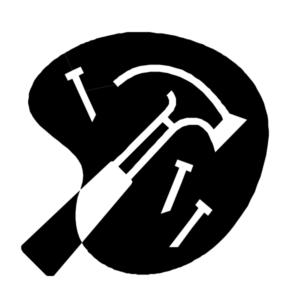
```
procedure TMainFrm.sFormClick(Sender: TObject);
begin
 If sForm.Checked Then
  MainFrm.FormStyle := fsStayOnTop Else
   MainFrm.FormStyle := fsNormal;
end;

end.
{********************************************* Coded By TF6M}
                        //15-07-2010
                        //Delphi4Arab
{********************************************* Coded By TF6M}
```

الملفات المصدرية مرفقة مع عدد المجلة.

تمرين هذا العدد

برمجة أداة Alias Manager
و
مكون Tables Manager



المطلوب:

برمجة أداة تسمح لنا بتسيير الـ Alias و تنظيم مسارات ملفات قواعد البيانات: إضافة، تعديل و حذف.

برمجة مكون يضاف إلى أي مشروع، دوره تعطيل أو تفعيل أي مكون من نوع Table.

بالتوفيق إن شاء الله